REMARKS

Claims 1-20 are pending in the present application. No claims were canceled; no claims were amended; and no claims were added. Reconsideration of the claims is respectfully requested.

Applicant would like to thank the Examiner for his courtesy in holding a telephone conference with Applicant's representative on March 10, 2005. During the telephone conversation, the Examiner and Applicant's representative discussed the patentability of the currently pending claims in light of *Aharon* (U.S. Patent No. 5,202,889). Applicant's representative remarked that *Aharon* only mentions drivers in a general discussion regarding drawbacks with the prior art. In conclusion, Applicant's representative added that the driver's conditions discussed in *Aharon*, such as processor priority level and memory addressing mode, are not equivalent to the present invention's drive states or wait states that transition randomly from state to state.

I. 35 U.S.C. § 103, Claims 1, 10, and 19

The Examiner has rejected claims 1, 10, and 19 under 35 U.S.C. § 103(a) as being unpatentable over *Aharon* (U.S. Patent No. 5,202,889) in view of "Exploiting Hardware Sharing in High-Level Synthesis for Partial Scan Optimization," *Dey*, IEEE 1063-6757/93, IEEE 1993. This rejection is respectfully traversed.

With regard to claim 1, the Examiner states:

> Regarding independent claim 1: Aharon discloses a method and system for generating pseudo random test patterns (CL 1-L21-31) for producing simulated test scenarios against a hardware model (CL 1-L51-61). Aharon discloses the elements of the claimed limitations of the present invention as follows:
>
> - generating driver model having states where each state indicates whether to drive an interface of hardware model: Aharon discloses test programs (patterns) that are simulated against a hardware model under driver control (CL2-L46) where the drivers can change the conditions with which the test programs are executed (CL2-L49). That is, the drivers disclosed by Aharon have "states" that indicate how, or how not to, "drive" the hardware model based on a set of conditions that determine the driver's current state. The examiner interprets applicant's driver model to be functionally equivalent to the driver control process disclosed by Aharon.
> - controlling simulation of hardware model using driver test pattern: Aharon discloses generating pseudo random test patterns

(CL 1-L21-31) under driver control (CL2-L47) of a simulated design model (CL 1-L60). That is, the test patterns are simulated against a hardware design model (CL2-L46), while the drivers controlling the test patterns (CL2-L48) can change the test conditions under which the test patterns are executed (CL2-L48), based on the current state of the driver (CL2-L47).
Aharon does not explicitly disclose using a random walk through the model to generate the driver test pattern.
- initiating random walk through driver model to generate driver test pattern: Dey teaches using a random walk technique (page 23, col. 2, para 4, Section 4.1) in the modeling of scan variables (test vectors) used for gate level hardware testing. The examiner notes that techniques such as random walks, table walks, walking bits, etc. are generally well-known to those skilled in the art and, hence, would have been an obvious choice for implementing the walk through the drive model, in addition to being taught by Dey. (See: Dey, Section 4.1)

It would have been obvious to one of ordinary skill in the art at the time the claimed invention was made to modify the teachings of Aharon relating to generating pseudo random test patterns against a hardware model, with the teachings of Dey relating to using a random walk technique on the driver model, to realize the claimed invention. An obvious motivation exists since, as referenced in the prior art, random selection of test patterns (instructions) is important in generating a sequence of test patterns because of the high probability selecting the same pattern (instruction) during the generation of a test pattern sequence (See Aharon, CL7-L49-67). Accordingly, a skilled artisan having access to the teachings of Aharon and Dey would have knowingly modified the teachings of Aharon with the teachings of Dey, in order to improve the randomness of the generation of the driver test patterns, and provide a more exhaustive test pattern sequence.

(*Office Action* dated December 13, 2004, pages 3-5). Independent claim 1, which is representative of independent claims 10 and 19, reads as follows:

1. A method for generating pseudo random test patterns for simulating a hardware model comprising:
generating a driver model having a plurality of states, wherein each state indicates whether to drive an interface of the hardware model;
initiating a random walk through the driver model to generate a driver test pattern; and
controlling simulation of the hardware model using the driver test pattern.

The Examiner bears the burden of establishing a prima facie case of obviousness based on prior art when rejecting claims under 35 U.S.C. § 103.. *In re Fritch*, 972 F.2d

1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). For an invention to be prima facie obvious, the prior art must teach or suggest all claim limitations. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). The *Aharon* and *Dey* references cited by the Examiner do not render obvious the present invention as recited in independent claims 1, 11, and 21 because the references fail to teach or suggest all claim limitations.

These rejected independent claims all recite generating a driver model having a plurality of states. Claim 1 recites "generating a driver model having a plurality of states, wherein each state indicates whether to drive an interface of the hardware model." Generating a driver model having a plurality of states is not a feature not taught or suggested in *Aharon*, *Dey* or any of the other references. *Aharon* is directed towards generating test patterns, but not generating a driver model having a plurality of states, as recited in claim 1 of the present invention. In the Office Action, the Examiner alleges that in the following cited passage *Aharon* teaches generating a driver model having a plurality of states.

> assembled test programs are simulated against an existing hardware or software model under driver control. These drivers can change the conditions with which the test programs are executed, e.g. processor priority

(*Aharon*, col. 2, lines 46-49).

Aharon teaches the use of "driver control" with which the "drivers can change the conditions with which the test programs are executed," but not "generating a driver model having a plurality of states." (*Aharon*, col. 2, lines 46-49; Applicant's Claim 1). Although *Aharon* teaches "conditions with which the test programs are executed," *Aharon* does not teach "a plurality of states" for the driver model. (*Id.*)

While *Aharon* does mention drivers in the passage cited by the Examiner, this reference is made only within the context of a general discussion regarding drawbacks with the prior art. After the written description covers the stated objectives of the invention, the lone occasion when the words "driver" or "drivers" appears is once in the discussion of instruction biasing found at column 8, lines 36 to 42. *Aharon* obviously does not teach generating driver models because it does not discuss driver models or drivers in the written description of the invention itself.

Although *Aharon* teaches that the prior art includes techniques wherein drivers can change the conditions with which the test programs are executed, the conditions are not equivalent to a plurality of states for a driver model. The intended meaning for the changed conditions cited by *Aharon* becomes clear when the full sentence containing the reference is examined. "These drivers can change the conditions with which the test programs are executed, e.g. processor priority level, memory addressing mode, etc." (Aharon, col. 2, lines 48-50). The alleged states in *Aharon* are actually priority levels and memory addressing modes.

In contrast, the states in the present invention are illustrated in Figure 4. For example, "state **402** is a 'drive' state, which dictates that the controlling code drive an interface of the hardware model," a drive state with "a probability of .01 or 1% of transitioning to state **404**, which is also a drive state," and "a probability of .99 or 99% of returning to state **402**" as part of a Markov chain where "the test pattern will likely include a 'burst' having at least five drive states and more likely having many more than five drive states." (Specification, pages 9-10). The present invention also has examples of a "wait state" that indicates "that the interface is not to be driven while in this state. Wait state **456** has a .01 probability of transitioning to state **458**" as part of a Markov chain that "will likely result in a 'stride' having at least two drive states followed by at least three wait states, although due to the probabilities each stride will more likely have many drive and wait states." (Specification, page 11). Therefore, the present inventions' states indicate whether the controlling code drives an interface of the hardware model, not the priority level or memory addressing mode from *Aharon*.

The Examiner further states that it would be obvious to combine *Aharon* with *Dey* to arrive at the features of the claimed invention. However, *Dey* does not cure the deficiencies in *Aharon*. *Dey* does not teach or suggest generating a driver model having a plurality of states, as recited in claim 1 of the present invention.

Applicant agrees with the Office Action that *Aharon* does not disclose using a random walk through model to generate the driver test pattern. Although *Dey* does teach a random walk-based approach, *Dey* does not teach or suggest generating a driver model having a plurality of states, as recited in claim 1 of the present invention. *Dey's* only mention of a random walk is brief.

> We refer to the above problem [selecting scan variables] as the minimum hardware-shared cut (HSC) problem. We address the minimum HSC problem by using a random walk-based approach which combines probabilistic and heuristic techniques. We use two measures, the loop cutting effectiveness measure, and the hardware sharing effectiveness measure, to capture the effectiveness of a variable in satisfying the three criteria [breaking CDFG loops, minimizing scan registers, and maximizing reuse of scan registers] of the minimum HSC problem.

(*Dey*, page 23, Section 4.1, column 2, paragraph 4)

Dey actually teaches the more limited application of a random walk-based approach to traverse edges in a data path in order to break loops and minimize the number of scan registers used for testing. In contrast, the present invention teaches a random walk for an entirely different purpose, as demonstrated by the following section.

> Next, the process connects the subgraphs (step **504**) and initiates a random walk through the Markov chains (step **506**). . . Once the driver module creates and connects the subgraphs of the Markov models, the driver module initiates a random walk through the Markov chains and provides the commands to the controlling code.

(Specification, page 11, lines 24, to page 12, line17)

The present invention uses a random walk to determine whether or not to transition from state to state in the Markov chain, with each state indicating whether to drive an interface of the hardware model. Therefore, the present invention uses a random walk to determine when the process is in a drive state and when the process is in a wait state, creating the required bursts and strides, as opposed to Dey's use of a random walk to traverse edges in order to break loops and minimize the use of scan registers.

Even if *Aharon* were combinable with *Dey*, the result of such a combination would not be the invention as recited in independent claim 1. Rather, such an alleged combination would result in a system for initializing required facilities for the execution of assembly language instructions, executing the instructions, and comparing the results system, substantially as taught in *Aharon*, with a random walk approach to minimize the number of scan registers in a data path, in the manner described by *Dey*. Even with the alleged additions of *Aharon* and *Dey*, there would be no ability for generating a driver model having a plurality of states, as recited in claim 1 of the present invention.

Furthermore, one of ordinary skill in the art would not combine *Aharon* with *Dey* when each reference is considered as a whole. In considering a reference as a whole, one of ordinary skill in the art would take into account the problems recognized and solved. *Aharon* is directed towards overcoming "the drawbacks of the conventional static approach to test pattern generation." (*Aharon*, col.3, lines 29-31). To achieve this purpose, *Aharon* lists its stated objectives:

> to provide a hardware verification process which permits its user (a designer and/or test engineer) to control the test pattern generation up to any required degree – from complete or partial specification of test patterns to completely automatic test pattern generation.
>
> It is also an object of this invention to relieve its user, while generating test patterns, from:
>
> initialization of required facilities (primary inputs, registers, memory elements and storage locations) which influence explicitly or implicitly the execution of the generated test pattern;
>
> tracing and marking all facilities which undergo changes during the execution of the generated test pattern and their comparison against the simulation results;
>
> calculating the test case's expected results.
>
> It is also an object of this invention to permit its user to generate test patterns for verification of a processor or entire computer system without detailed knowledge of the processor assembly language.

(*Aharon*, col.3, lines 32-52). Thus, *Aharon* is directed towards generating test patterns based upon the processor assembly language; initializing primary inputs, registers, memory elements and storage locations; executing the generated test pattern; and comparing the simulation results.

In contrast, *Dey* is directed towards exploiting hardware sharing to minimize the number of scan registers needed to synthesize a testable data path. *Dey* proposes algorithms to select a minimum number of scan registers to cut CDFG loops, and reuse the scan registers during scheduling and assignment to avoid the formation of further loops in the data path. (*Dey*, Section 7, Conclusions). Thus, *Dey* minimizes the number of scan registers in a data path in order to reduce the number of loops in the data path.

In view of the above, there is no motivation to combine the teachings of *Aharon* with *Dey* in the manner alleged by the Examiner. The Examiner alleges that the motivation for the alleged combination is "to improve the randomness of the generation of driver test patterns, and provide a more exhaustive test pattern sequence." However,

there is no suggestion in *Aharon* that there is a need to improve the randomness of the generation of test patterns. In fact, *Aharon* is directed toward initializing required facilities for the execution of assembly language instructions, executing the instructions, and comparing the results. The test patterns in *Aharon* merely teach a user to verify a processor or entire computer system without detailed knowledge of the processor assembly language. There is no need, let alone any suggestion, to improve the randomness of the generation of test patterns in *Aharon*.

Moreover, there is no suggestion in *Dey* of a need to integrate the random walk approach of *Dey* with the initialization, execution and comparison of results for assembly language instructions, such as that taught by *Aharon*. *Dey* has nothing to do with generating a more exhaustive test pattern sequence. In fact, *Dey* is concerned with minimizing the number of scan registers so a data path can be tested more easily. The paragraph in *Dey* cited by the Examiner to teach a random walk-through technique actually teaches traversing a data path in order to minimize the assignment and allocation of scan registers. There is no need, let alone any suggestion in *Dey* to generate exhaustive test patterns. Thus, the alleged motivation offered by the Examiner is not based on the actual teaching of the references.

As noted above, there is no teaching or suggestion in the references as to the desirability of including the features from other references. The mere fact that a prior art reference can be readily modified does not make the modification obvious unless the prior art suggested the desirability of the modification. *In re Laskowski*, 871 F.2d 115, 10 U.S.P.Q.2d 1397 (Fed. Cir. 1989) and also *see In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992) and *In re Mills*, 916 F.2d 680, 16 U.S.P.Q.2d 1430 (Fed. Cir. 1993). The Examiner may not merely state that the modification would have been obvious to one of ordinary skill in the art without pointing out in the prior art a suggestion of the desirability of the proposed modification. The Examiner has failed to demonstrate any motivation or incentive in the prior art to combine and modify the references so as to achieve the claimed invention. The only motivation to even to attempt to combine *Aharon* and *Dey* is to try to arrive at Applicant's claimed invention and thus, the alleged combination is a result of impermissible hindsight reconstruction using Applicant's own disclosure as a guide. While Applicant understands that all examination entails some measure of hindsight, when the rejection is based completely on hindsight,

as in the present case, rather than only what is gleaned from the references by one of ordinary skill in the art, then the rejection is improper and should be withdrawn.

Moreover, neither *Aharon* nor *Dey* teaches the problem of the present invention or its source. The present invention recognizes the problem of purely random patterns failing to test the limits of the system being simulated. To compensate for this shortcoming, the present invention generates test patterns that have the desired bursts and strides. Thus, the present invention provides the benefits of generating pseudo random test patterns without the problems associated with purely random patterns. Therefore, one of ordinary skill in the art would not be motivated to modify *Aharon* and *Dey* in the manner required to form the solution discussed in the claimed invention when the problems addressed by the two references are reviewed when considering each reference as a whole.

In view of the above, Applicant submits that independent claims 1, 10, and 19 are not taught or suggested by the alleged combination of *Aharon* and *Dey*. Accordingly, Applicant respectfully requests withdrawal of the rejection of independent claims 1, 10, and 19 under 35 U.S.C. §103.

Claims 2-9 are dependent claims depending from independent claim 1. Claims 11-18 are dependent claims depending from independent claim 10. Claim 20 is a dependent claim depending from independent claim 19.Applicant has already demonstrated claims 1, 10, and 19 to be in condition for allowance. Applicant respectfully submits that claims 2-9, 11-18, and 20 are also allowable, at least by virtue of their dependency on an allowable claim.

II. 35 U.S.C. §103, Alleged Obviousness, Claims 2 and 11

The Examiner has rejected claims 2 and 11 under 35 U.S.C. § 103 (a) as being unpatentable over *Aharon* (U.S. Patent No. 5,202,889) in view of "Exploiting Hardware Sharing in High-Level Synthesis for Partial Scan Optimization," *Dey*, IEEE 1063-6757/93, IEEE 1993. This rejection is respectfully traversed.

In regards to claims 2 and 11, the Office Action alleges that *Aharon* discloses:

> controlling test patterns by driver state as noted above (CL2-L46-49). Aharon further discloses the use of "loop" logic in "waiting" to obtain the desired sequences for test patterns. (i.e. an equivalent function to wait states) The examiner notes that the use of "wait states" is very well

known in the art as a way of having a process "wait" for data results (See: "wait state", Microsoft Dictionary, Third Edition, 1997).

(*Office Action*, dated December 3, 2004, page 5 for claim 2, pages 7-8 for claim 11)..

Claim 2, which is representative of dependent claim 11, specifies that each state of the plurality of states in claim 1 comprises one of a drive state and a wait state. As discussed above in regards to independent claims 1 and 10, *Aharon* does not teach generating a driver model having a plurality of states, as recited in independent claims 1 and 10, upon which dependent claims 2 and 11 are dependent, respectively. Therefore, *Aharon* does not disclose generating a plurality of states comprised of drive states and wait states.

Additionally, Claim 2 is a dependent claim depending on independent claim 1and claim 11 is a dependent claim depending on independent claim 10. Applicant has already demonstrated claims1 and 10 to be in condition for allowance. Applicant respectfully submits that claims 2 and 11 are also allowable, at least by virtue of their dependency on allowable claims.

III. 35 U.S.C. §103, Alleged Obviousness, Claims 3 and 12

The Examiner has rejected claims 3 and 12 under 35 U.S.C. § 103(a) as being unpatentable over *Aharon* (U.S. Patent No. 5,202,889) in view of "Exploiting Hardware Sharing in High-Level Synthesis for Partial Scan Optimization," *Dey*, IEEE 1063-6757/93, IEEE 1993, in further view of *Ashar* (U.S. Patent No. 6,163,876). This rejection is respectfully traversed.

In regards to claims 3 and 12, the Examiner alleged that:

> As recited above, the combination of Aharon and Dey renders obvious the elements of the claimed limitations of independent claims 1 and 10. (See rejections of claims 1 and 10 above) However, the combination of Aharon and Dey further does not explicitly teach the use of a sub-graph connecting the driver model as recited in dependent claims 3 and 12.
>
> Per dependent claims 3 and 12 – (means for) creating driver sub-graph having states & connecting sub-graph to form driver model: Ashar teaches the use of sub-graphs having multiple states (CL9-L30, L61-65, CL10-L4, Fig 1b) in the verification and testing of a hardware model (CL5-L11, CL10-L12-17) and connecting the sub-graphs (Fig. 1b) according to state. The examiner notes that, in addition to being disclosed by Ahser, sub-graphs are merely a

> subset of the nodes and edges of a well-known graph data structure (See: "graph (subgraph)", Microsoft Dictionary, Third Edition, 1997) and, hence would have been an obvious choice to one skilled in the art for connecting the driver model at the time of the invention. Therefore, it would have been obvious to one of ordinary skill in the art at the time the claimed invention was made to further modify the combined teachings of Aharon and Dey as previously noted above, with the teaching of Asher relating to connecting the sub-graphs in forming the driver model, in order to improve the randomness of the generation of the driver test patterns, and provide a more exhaustive pattern sequence.

(*Office Action*, dated December 13, 2004, pages 8-9).

Applicant agrees with the Office Action that the combination of *Aharon* and *Dey* does not disclose the use of a sub-graph connecting the driver model as recited in dependent claims 3 and 12. Claim 3, which is representative of dependent claim 12 with respect to similarly recited subject matter, specifics "creating at least one driver subgraph having a plurality of states."

As discussed above in regards to independent claims 1 and 10, the combination of *Aharon* and *Dey* does not teach generating a driver model having a plurality of states, as recited in independent claims 1 and, upon which dependent claims 3 and 12 are dependent, respectively. Although *Ashar* teaches at least one subgraph that has a plurality of states, *Ashar* is directed towards verifying register-transfer logic against its scheduled behavior in a high-level synthesis environment. (*Ashar*, Abstract). But because *Ashar* does not discuss or even mention drivers, *Ashar* does not provide for the deficiencies in the combination of *Aharon* and *Dey*. Therefore, the combination of *Aharon, Dey* and *Ashar* does not teach generating a driver model having a plurality of states wherein at least one driver subgraph has a plurality of states.

Claim 3 is a dependent claim depending on independent claim 1, and claim 12 is a dependent claim depending on claim 10. Applicant has already demonstrated claims1 and 10 to be in condition for allowance. Applicant respectfully submits that claims 3 and 12 are also allowable, at least by virtue of their dependency on allowable claims.

IV. 35 U.S.C. §103, Alleged Obviousness, Claims 4, 5, 13, and 14

The Examiner has rejected claims 4, 5, 13, and 14 under 35 U.S.C. § 103(a) as being unpatentable over *Aharon* (U.S. Patent No. 5,202,889) in view of "Exploiting

Hardware Sharing in High-Level Synthesis for Partial Scan Optimization," *Dey*, IEEE 1063-6757/93, IEEE 1993, in further view of *Ashar* (U.S. Patent No. 6,163,876) and in further view of Gil (U.S. Patent No. 5,500,941). This rejection is respectfully traversed.

The Office Action alleges that

> As recited above, the combination of Aharon, Dey and Asher renders obvious the elements of the claimed limitations of independent claims 1 and 10 and dependent claims 3 and 12. (see rejection of claims 1, 10, 3, and 12 above) However, the combination of Aharon, Dey, and Asher further does not explicitly teach the use of a Markov chain or probability of transitioning between states as recited in dependent claims 4, 13 and 5, 14 respectively.
>
> Per dependent claims 5 and 14 – probability of state transitioning: Gil discloses calculating the probabilities of the occurrence of state transitions from one state to another (CL4-L55, CL6-L38-40, Fig. 4).
>
> Per dependent claims 4 and 13 – Markov chain: Gil discloses the use of Markov chains for generating state transition using during testing. (CL4-L47-56, Fig. 1)
>
> The examiner again notes that, in addition to being disclosed by Gil, a Markov chain is merely "a random process where the probability that certain state will occur depends only on the present or preceding state of the system, and not the events leading up to the present state". (Encyclopedia of Computer Science, Mason/Charter, 1976) Markov chains are well known to those skilled in the art and are commonly used as a method of generating random samples from a probability space and, hence, would have been an obvious choice to one skilled in the art for implementing in the driver sub-graph. Therefore, it would have been obvious to one of ordinary skill in the art at the time the claimed invention was made to further modify the combined teachings of Aharon, Dey, Asher, as previously noted above, with the teaching of Gil relating to Markov chain probability, in order to improve the randomness of the generation of driver test patterns, and provide a more exhaustive pattern sequence.

(*Office Action*, dated December 13, 2004, pages 9-10).

Applicant agrees with the Office Action that the combination of *Aharon*, *Dey*, and *Ashar* does not teach that the use of Markov chains or probability transitioning between states. Claim 4, which is representative of dependent claim 13 with respect to similarly recited subject matter, specifies "wherein each driver subgraph is a Markov chain." Claim 5, which is representative of dependent claim 14 with respect to similarly recited subject matter, specifies "wherein each state has a probability of transitioning to at least one other state."

As discussed above in regards to independent claims 1 and 10, the combination of *Aharon*, *Dey*, and *Ashar* does not teach or suggest generating a driver model having a plurality of states, as recited in independent claims 1 and 10, upon which dependent claims 4-5 and 13-14 are dependent, respectively. Although *Gil* teaches the use of Markov chains or probability transitioning between states. *Gil* is directed towards